

A GENERALIZED ASYNCHRONOUS COMPUTABILITY THEOREM

ELI GAFNI¹, PETR KUZNETSOV², AND CIPRIAN MANOLESCU³

ABSTRACT. Two decades ago, the celebrated Asynchronous Computability Theorem (ACT) characterized the class of wait-free solvable distributed tasks by casting the task solvability question to the existence of a continuous and carrier-preserving map between simplicial complexes. However, the theorem did not extend to other models, such as the models of t -resilience or obstruction-freedom. Task solvability in these models has been studied in an *ad hoc* manner, via model reductions or carefully crafted operational interpretations.

In this paper, we derive a general topological characterization of solvability that applies to any subset of runs of the wait-free model. After getting all the right definitions in place, the theorem itself is surprisingly easy to derive.

We demonstrate the utility of our characterization by considering the t -resilient solvability of a task, whose direct operational solution is quite involved. In contrast, our generalized computability theorem determines the solvability in a straightforward manner.

1. INTRODUCTION

This paper characterizes task solvability in models of distributed computing, where processes communicate via reading from and writing to shared registers. In this context, a model is a set of *runs*, i.e., interleavings of read and write steps issued by different processes.

What do we mean by a characterization? We say that a task T is solvable in a model M , if there exists a *protocol* by which, in every run of M , each process taking sufficiently many steps eventually *outputs*, so that the outputs satisfy the task's specification with respect to the provided inputs. Thus, solvability is defined *operationally*, via the existence of a protocol. Giving a characterization means reducing the operational definition to the existence of a map between topological spaces, capturing the sets of possible input and output configurations of the task. Hopefully, this topological characterization for a given instance should provide insights that are not easy to grasp operationally.

In 1993, Herlihy and Shavit [20, 21] characterized read-write communication with no restrictions on the interleaving, referred to as the *wait-free* model. They formulated the Asynchronous Computability Theorem (ACT) stating that a task T is wait-free solvable *if and only if* there exists a simplicial map from a subdivision of simplexes of an appropriately defined input simplicial complex to an appropriately defined output simplicial complex. The original formulation of ACT is done for the *standard shared-memory* model, where memory consists of persistent objects which can be written and read arbitrarily often. In [3], Borowsky and the first author came out with a surprisingly easy re-derivation of

¹COMPUTER SCIENCE DEPARTMENT, UCLA, 3731F BOELTER HALL, LOS ANGELES, CA 90095, USA

²TELEKOM INNOVATION LABORATORIES/TU BERLIN, SEKR. TEL 16, ERNST-REUTER-PLATZ 7, 10587 BERLIN, GERMANY, AND TÉLÉCOM PARISTECH, 46 RUE BARRAULT, 75013 PARIS, FRANCE

³DEPARTMENT OF MATHEMATICS, UCLA, 520 PORTOLA PLAZA, LOS ANGELES, CA 90095, USA

E-mail addresses: eli@ucla.edu, petr.kuznetsov@tu-berlin.de, cm@math.ucla.edu.

CM was supported by NSF grant DMS-1104406.

ACT for the *iterated immediate snapshot* model (IIS); using simulation, they proved there that IIS is equivalent to the standard shared-memory model with respect to solving tasks. The explanation of this simplicity is that there are transformations which are best suitable to the operational treatment (simulating one model by the other), and others which are best suitable to the topological treatment (bringing to bear “global” arguments). Indeed, in [3], the bulk of the complexity of the ACT derivation was subsumed by the algorithmic simulation that showed the operational equivalence between the standard model and IIS.

Here, we follow the footsteps of [3] in using IIS to generalize ACT to *any* restricted IIS model, i.e., a model consisting of only some of the runs in the wait-free IIS model. As mentioned, IIS is natural for deriving topological results as it allows for an intuitive topological representation as a standard chromatic subdivision of the input complex [22, 23] that is tricky to derive in the standard model [21]. An additional incentive to focus on the IIS models is because it is richer than the standard ones in the sense that they exhibit interesting interleavings that cannot be observed in the standard models (see Section 3.4 for more detail).

ACT turned out to be an essential tool in distributed computing [5, 12, 16–19]. We show that our *generalized* asynchronous computability theorem (GACT) holds that promise, too. We consider a task T , solvable t -resiliently, but (to our knowledge) with a very involved algorithm. (The reader is invited to the challenge of finding this algorithm, or better yet, a simpler one.) In contrast, by applying GACT, we show that determining the t -resilient solvability of T is immediate (modulo certain technicalities of going from a point-set continuous map to a simplicial, color-preserving one). The complexity of the algorithm is hidden in the choice one makes in GACT, as GACT talks about the existence of a “terminating subdivision.” Once the correct choice of subdivision is made, the existence of the algorithm is apparent.

There is a fundamental difference between the wait-free model and the t -resilient model. The wait-free model is *compact*: in a topology proposed in [1] and defined precisely later in this paper, any sequence of runs in the model attains a convergent subsequence whose limit is in the model. An algorithm to solve a task in a compact model provides outputs in all runs, in a bounded number of steps. This claim does not hold for non-compact models, such as the model of t -resilience with $0 < t < n$ (in a system of $n + 1$ processes).

Checking task solvability in non-compact models required all sort of roundabouts, starting with the BG-simulation [2, 4], and continuing in [11]. The latter uses the method by which tasks were solved t -resiliently over the years, of reducing the t -resilient model to an equivalent model which is compact. Yet, all these techniques are limited and the reduction is *ad hoc* in nature, i.e., it must be carefully adjusted to each model of computation. In fact, as a by-product of GACT, we derive a topological characterization of the t -resilient model, a result claimed by Herlihy and Shavit in [20] but not pursued later in [21].

To derive a direct characterization of any (possibly non-compact) model, without an *ad hoc* reduction to a compact one, it is natural to represent the model using infinite simplicial complexes. Indeed, the topological characterization that we give in GACT is in terms of simplicial maps from an infinite complex to (finite) output complexes.

We should note that most of the technical work in this paper is definitional, setting the grounds for the surprisingly easy derivation of GACT. It resolves many ambiguities in the definitions that appear in the literature on task solvability. In particular, it resolves the confusion of whether a process *halts* after delivering an output (it does not!). It also gives a universal definition of task solvability across models, based only on the provided notions

of a step and a view in the model. Previously, the notion of “step” in IIS was not clear and was usually thought about through the simulation in [3, 13]; this approach proved too limiting and inelegant for our current work.

The paper is organized as follows: In Section 2 we describe the IIS model and give some examples of restricted IIS models. In Section 3 we review the topological definition of a task, and explain what it means for a task to be solvable in a model. In Section 4 we describe restricted IIS models topologically. In Section 5 we prove our main result, GACT. In Section 6 we explain how GACT gives back the well-known ACT in the wait-free case. Finally, in Section 7 we use GACT to show that a particular task can be solved in the t -resilient model, and in Section 8 we draw the conclusions.

2. MODELS

In this section, we describe our perspective on the *Iterated Immediate Snapshot* (IIS) model [3]. There is no loss of generality in using the IIS model, rather than the standard read-write shared-memory model [15], since there is a one-to-one correspondence between the runs of the standard model and the runs of the iterated one [3, 13]. We need to present “our” perspective since the notion of a *participating set* for IIS was never defined precisely (only through the simulation in [3, 13]) and, consequently, the notion of task solvability in IIS was tackled until now only informally.

Distributed computing is about the interleaving of actions taken by different processes. Possible interleavings of primitive steps of the processes are naturally described by sequences of the process identifiers. In the *immediate snapshot* (IS) model [2], a primitive step is for a set of processes to simultaneously write and then simultaneously read. Thus, an interleaving of actions in IS is captured by a sequence of sets of process ids. In the presence of a single shared-memory this would give rise to the model called *repeated immediate snapshot* (RIS) [2, 25].

The iterated immediate snapshot model (IIS) further stratifies writing and reading, by assuming the existence of an infinite sequence of immediate snapshot memories MEM_1, MEM_2, \dots , and the k^{th} appearance of p_i in IIS is to be interpreted as its writing to and reading from MEM_k . Thus, processes in the same set in an interleaving might write to and read from different memories.

Formally, the IIS model on $n + 1$ processes p_0, \dots, p_n consists of *runs*, where each run is an infinite sequence of sets of process ids. Let $r = S_1, S_2, \dots$ be a run. If there is a k such that $p_j \in S_k$ then p_j is *participating* in r . If p_j appears in infinitely many sets of r , we say that p_j is *infinitely participating* in r . The sets of participating and infinitely participating processes in a run r are denoted $part(r)$ and $\infty\text{-}part(r)$, respectively.

Fix a run $r = S_1, S_2, \dots$. For each S_k , the k^{th} step in r , and $p_j \in S_k$, we define $layer(p_j, S_k) = m$ if the appearance of p_j in S_k in r is the m^{th} appearance of p_j in r ; i.e., if at the k^{th} step in the run, the process p_j writes and reads in MEM_m . For convenience, we often add superscripts to the process ids in r according to their layers, i.e., p_j in S_k is written p_j^m .

Following the interpretation, we define an equivalence relation between sequences in IIS. If $p_j^m, p_i^q \in S_k$ are such that $m \neq q$, then in S_k , the processes p_j and p_i write to and read from different memories. Thus we can partition S_k (at its place in the sequence) according to common superscripts. After this, sets are associated with a single superscript and we can define $layer(S_k)$ to be the unique superscript of the processes in it. We can then commute

S_k and S_{k+1} if $\text{layer}(S_{k+1}) < \text{layer}(S_k)$. We declare that a run is equivalent to the new run obtained by doing these moves.

A run in which each set is of a single superscript and small superscript set precedes a higher superscript set is called *canonical*, representing its equivalence class. For example, a run that starts with

$$S_1 = \{p_0^1, p_1^1, p_2^1\}, S_2 = \{p_0^2, p_1^2\}, S_3 = \{p_1^3\}, S_4 = \{p_0^3, p_2^2\}, \dots$$

is equivalent to a canonical run that starts with

$$S'_1 = \{p_0^1, p_1^1, p_2^1\}, S'_2 = \{p_0^2, p_1^2\}, S'_3 = \{p_2^2\}, S'_4 = \{p_1^3\}, S'_5 = \{p_0^3\}, \dots$$

Without loss of generality, we will restrict our attention to the set \mathcal{R} of canonical runs.

Given a run $r \in \mathcal{R}$, we define a set called the *view* of p_i^k in r recursively:

- (1) $\text{view}(p_i^0, r) = \{p_i\}$;
- (2) For $k \geq 1$, the view of $p_i^k \in S_m$ is $\text{view}(p_i^k, r) = \{\text{view}(p_j^{k-1}, r) \mid p_j^k \in S_q, q \leq m\}$.

The minimal participating set $\text{min-part}(r)$ of a run $r \in \mathcal{R}$ is defined to be the minimal set of processes such that for each $p_i \in \infty\text{-part}(r)$, the views of p_i in r and in the restriction $r|_{\text{min-part}(r)}$ are the same. Note that $\text{min-part}(r)$ contains $\infty\text{-part}(r)$.

The set of *fast processes* in a run r is defined as the minimal set $\text{fast}(r) \subseteq \{0, 1, \dots, n\}$ such that after a certain step, the processes in $\text{slow}(r) := \{0, 1, \dots, n\} \setminus \text{fast}(r)$ (called the *slow processes*) have no new appearance in the view of the processes in $\text{fast}(r)$.

2.1. Examples of models. We define a *restricted IIS model* M to be any subset of \mathcal{R} .

Example 2.1. The *wait-free* (or *completely asynchronous*) model WF is the set \mathcal{R} itself. The interpretation of WF is that anything can happen (all sorts of step interleavings are allowed).

Example 2.2. For $t \leq n$, the *t-resilient model* Res_t consists of the runs $r \in \mathcal{R}$ such that $|\text{fast}(r)| \geq n + 1 - t$. This is the model in which at most t processes are slow.

Example 2.3. For $t \leq n + 1$, the *t-obstruction-free model* OF_t consists of all the runs r with $|\text{fast}(r)| \leq t$. This model was previously discussed in [10], following a suggestion of Guerraoui. It is the model in which no more than t processes are fast.

Example 2.4. More generally, consider the *model with adversary* \mathbb{A} [6], which we denote by $M^{\text{adv}}(\mathbb{A})$. Here, \mathbb{A} is any subset of the power set of $\{0, 1, \dots, n\}$. We then define $M^{\text{adv}}(\mathbb{A})$ to consist of all runs r such that $\text{slow}(r) \in \mathbb{A}$.

3. TASKS

3.1. Definitions. To reduce clutter we focus on *input-less* tasks, i.e. tasks where the input to each process in $\{p_0, \dots, p_n\}$ is its own id (also called *color*). Extending GACT to tasks with multiple inputs is straightforward.

Let us start with a few topological definitions. We let \mathbf{s} denote the standard n -simplex with $n + 1$ vertices. A *chromatic complex* is an abstract simplicial complex C together with a simplicial map $\chi : C \rightarrow \mathbf{s}$ (the coloring map). Every chromatic complex C has a geometric realization $|C|$. It also has *standard chromatic subdivision* $\text{Chr } C$. This is defined inductively on the dimension of C . If C is zero dimensional, we let $\text{Chr } C = C$. Suppose C has dimension n , and that we already took the chromatic subdivision of its $(n - 1)$ -skeleton. For each n -simplex τ in C , take a new n -simplex τ' . For each face \mathbf{t} of τ , let $\bar{\mathbf{t}}'$ be the complementary face of τ' (that is, the one colored by the colors in the complement of the

colors of \mathbf{t}). Then cone off $\bar{\mathbf{t}}'$ with all the simplices in the chromatic subdivision of \mathbf{t} . The resulting complex is $\text{Chr } C$. That $\text{Chr } C$ is a subdivided simplex was proved in [22, 23]. If we iterate this construction m times we obtain the k^{th} chromatic subdivision, $\text{Chr}^k C$.

A *chromatic map* between chromatic complexes A and B is a simplicial map that respects the colors. A *chromatic multi-map* between A and B is a map $\Delta : A \rightarrow 2^B$ that, for any $m \leq n$, takes every m -simplex of A to a non-empty, pure m -dimensional subcomplex of B , such that: (i) For any simplex σ of A , we have $\chi(\sigma) = \chi(\Delta(\sigma))$, and (ii) For any simplices $\sigma, \tau \in A$, we have $\Delta(\sigma \cap \tau) \subseteq \Delta(\sigma) \cap \Delta(\tau)$.

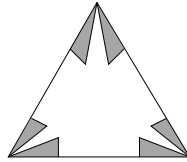
A *task* $T = (C, \Delta)$ on $n+1$ processes $\{p_0, \dots, p_n\}$ will be a pure n -dimensional chromatic complex C together with a chromatic multi-map $\Delta : \mathbf{s} \rightarrow 2^C$. C is called the *output complex*.

For $k > 0$, a non-empty complex C is *k-connected* if, for all $m \leq k$, any continuous map of the m -sphere on $|C|$ can be extended to a continuous map over the $(m-1)$ -disk.

3.2. Affine tasks. Many examples of tasks can be constructed as follows. Let $L \subseteq \text{Chr}^k \mathbf{s}$ be a pure n -dimensional subcomplex of the k^{th} chromatic subdivision of \mathbf{s} , for some k . Assume that for each face $\mathbf{t} \subseteq \mathbf{s}$, the intersection $L \cap \text{Chr}^k \mathbf{t}$ is a non-empty, pure subcomplex of $\text{Chr}^k \mathbf{s}$ of the same dimension as \mathbf{t} . We define a task (L, Δ) by setting $\Delta(\mathbf{t}) = L \cap \text{Chr}^k \mathbf{t}$ for any face $\mathbf{t} \subseteq \mathbf{s}$. Tasks constructed like this are called *affine*. To depict an affine task, we can simply draw the corresponding output complex.

By abuse of notation, we will usually write L for the affine task (L, Δ) . We chose the name *affine* because if we have a task L as above, the geometric realizations of the simplices of L can be depicted as lying on affine subspaces of \mathbb{R}^n . Similar terminology appears in algebraic geometry, where one talks about affine varieties.

For example, consider the task of *total order* T^{ord} , defined as follows. For each permutation α of $\{0, 1, \dots, n\}$, there is a unique n -simplex σ_α in the second chromatic subdivision $\text{Chr}^2 \mathbf{s}$ with the property that the vertex of σ_α colored i is in the interior of an i -dimensional face of \mathbf{s} . For example, for 3 processes, the six simplices of the form σ_α are those shown here:



The total order task is the affine task whose output complex $C^{\text{ord}} \subseteq \text{Chr}^2 \mathbf{s}$ is the union of all the $(n+1)!$ simplices of the form σ_α . The name total order refers to the fact that the possible outputs (when all $n+1$ processes are running) are in one-to-one correspondence with the total orderings (i.e., permutations) of the set $\{0, 1, \dots, n\}$.

3.3. Task Solvability. Any conceivable distributed computing model M on $\{p_0, \dots, p_n\}$ is a set of infinite sequences (runs) of primitive abstract *steps*, together with a recursive function *view* that takes a step in r and maps it to some output values, and is dependent only on the prefix of r before the step. Steps are associated with processes or group of processes.

In a restricted IIS model, informally, a task $T = (C, \Delta)$ is solvable in M if for all runs $r \in M$, the infinitely participating processes output, and their output is a subsimplex of the allowed outputs for the minimally participating processes. An output is the result of a *protocol* and for us, when dealing with solvability rather than complexity, a protocol is just a partial map from views to some output value range. Thus, requiring an infinitely

participating process to output means requiring that eventually it will have a view that is mapped by the protocol to an output value.

Formally, a protocol Π is a partial map from views to outputs.

Definition 3.1. A task $T = (C, \Delta)$ is *solvable* in a restricted IIS model M if there exists a protocol Π such that for all $r \in M$:

- (1) For each p_i , there exist k_0 and a vertex c of C colored i , such that:
 - For all $k < k_0$, $view(p_i^k, r) \notin domain(\Pi)$;
 - For all $k \geq k_0$ such that p_i^k exists, we have $\Pi(view(p_i^k, r)) = c$.
 (This condition is satisfied vacuously if p_i is not infinitely participating, because we can find k_0 such that p_i did not take k_0 steps in r , so p_i^k does not exist for $k \geq k_0$.)
- (2) For all k , $\{\Pi(view(p_i^k, r)) \mid view(p_i^k, r) \in domain(\Pi)\}$ is a simplex in $\Delta(min-part(r))$.

3.4. IIS versus the standard model. Consider a restricted IIS model M . We can define another model M_{fast} by replacing each $r \in M$ with trimmed run r' in which we eliminated from r all steps that leave views of processes in $fast(r)$ intact. If T is solvable in M then it is obviously solvable in M_{fast} , but not vice-versa. Indeed, consider the obstruction-free model $OF = OF_1$, consisting of runs with a single fast process. Obviously, OF cannot produce a total order between all ∞ -participating processes since in runs where the process in $fast(r)$ is in front from the beginning, the rest of the processes proceed wait-free and cannot solve total-order among themselves. On the other hand OF_{fast} solves total-order via Commit-Adopt [8].

We see from here that considering restricted IIS models is more refined than considering restrictions of the standard shared-memory model. Indeed, for any subset of runs M' of the standard shared-memory model, there is an equivalence between M' and some M_{fast} , where M is a restricted IIS model [3, 13]. These simulations were non-blocking simulations. To see that what we call t -resilient in IIS corresponds to t -resilience in the standard model one needs to additionally employ the reductions in [7]. However, we can find many restricted IIS models M with the same M_{fast} . Some of these models will require outputs by processes that are considered faulty (and hence not obligated to output) in M' . Thus, a task T that is solvable in M' might not be solvable by some of the IIS models M that map to M' through the simulation in [3, 13].

4. TOPOLOGICAL INTERPRETATION

Recall that \mathcal{R} denotes the set of (canonical) runs in IIS. We put a metric topology on \mathcal{R} as follows. Given runs $r, r' \in \mathcal{R}$, we let $k = k(r, r')$ denote the smallest $k \geq 0$ such that the first k steps of r and r' are identical. (In particular, we let $k = \infty$ when $r = r'$.) We set the distance between r and r' to be $d(r, r') = 1/(1 + k)$. For future reference, we mention:

Lemma 4.1. *The metric space \mathcal{R} is compact.*

Proof. Let $r^1 = (S_1^1, S_2^1, \dots)$, $r^2 = (S_1^2, S_2^2, \dots)$, \dots be an infinite sequence of runs. There are only finitely many possibilities for the first step S_1^j , so we can find a subsequence such that the first step is a constant choice S_1 . From it we can extract a further subsequence such that the second step is a constant choice S_2 , and so on. Using Cantor's diagonal method we get a subsequence that converges to the run $r = (S_1, S_2, \dots)$. \square

The topology on \mathcal{R} is rather hard to visualize. We can get a partial understanding by focusing on the views of the fast processes in each run.

Indeed, consider the standard chromatic subdivisions of the n -simplex \mathbf{s} . A run in IIS can be identified with a sequence of simplices $\sigma_0, \sigma_1, \sigma_2, \dots$, with $\sigma_k \in \text{Chr}^k \mathbf{s}$ and $|\sigma_{k+1}| \subset |\sigma_k|$.

Note that every run converges to a point of the geometric realization $|\mathbf{s}|$, so there is a natural, continuous map $\pi : \mathcal{R} \rightarrow |\mathbf{s}|$, which we call the *affine projection*. The information captured in $p = \pi(r)$ exactly consists of the views of the fast processes in r .

There is a canonical coloring map $\chi : |\mathbf{s}| \rightarrow 2^{\{0,1,\dots,n\}}$ that extends the colorings on all chromatic subdivisions $\text{Chr}^m \mathbf{s}$ to $|\mathbf{s}|$. Precisely, given a point $p \in |\mathbf{s}|$, we let $\chi(p)$ be the minimal subset $A \subseteq \{0, 1, \dots, n\}$ such that p lies in a simplex σ of a chromatic subdivision $\text{Chr}^k \mathbf{s}$ with $\chi(\sigma) = A$. It is easy to see that $\chi(p) = \text{fast}(r)$, for any r such that $\pi(r) = p$.

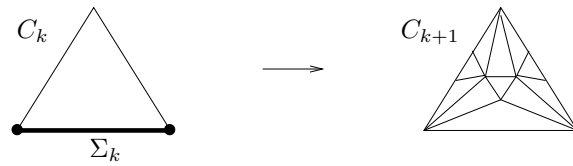
The space $|\mathbf{s}|$ has appeared previously in the work of Saks and Zaharoglou [25]. The points of this space induce a partition of the set of runs. The models in Examples 2.1-2.4 all obey this partition, i.e., they are of the form $\pi^{-1}(S)$, where $S \subseteq |\mathbf{s}|$ is a subset. We call models of the form $\pi^{-1}(S)$ *geometric*, because they can be easily visualized in terms of the associated subset of $|\mathbf{s}|$. However, our main results will apply equally well to *all* (not necessarily geometric) restricted IIS models.

5. THE GENERALIZED ASYNCHRONOUS COMPUTABILITY THEOREM

5.1. Terminating subdivisions. Let C be a chromatic complex. Recall from Section 3 that we have standard chromatic subdivisions $\text{Chr}^m C$ for all $m > 0$. Note that the vertices of $\text{Chr}^m C$ can be identified with a subset of the vertices in $\text{Chr}^{m+1} C$.

A *terminating subdivision* \mathcal{T} of C is specified by a sequence of chromatic complexes $C = C_0, C_1, C_2, \dots$, and a sequence of subcomplexes $\Sigma_0 \subseteq \Sigma_1 \subseteq \Sigma_2 \subseteq \dots$ such that:

- (i) Σ_k is a subcomplex of C_k ;
- (ii) C_{k+1} is obtained from C_k by taking the partial chromatic subdivision in which the simplices in Σ_k are “terminated”, i.e., not further subdivided. Precisely, we construct C_{k+1} from C_k inductively on the dimension of the skeleta, as in the definition of Chr^1 in Subsection 3.1, except that at each inductive step we introduce new simplices τ' only for those simplices τ that are not in Σ_k . For example, if Σ_k is zero-dimensional, then $C_{k+1} = \text{Chr}^1 C_k$; if C_k is the standard 2-dimensional simplex and Σ_k is one of its 1-dimensional faces, we have:



A simplex of Σ_k for some k is called a *stable simplex* in the subdivision \mathcal{T} ; such a simplex remains unchanged in all further complexes C_{k+1}, C_{k+2}, \dots . The stable simplices in \mathcal{T} form a chromatic complex, which we denote by $K(\mathcal{T})$; it usually has infinitely many vertices. Observe that the geometric realization $|K(\mathcal{T})|$ can be identified with a subset of $|C|$.

In particular, if there exists i such that $\Sigma_k = C_k$, then we must have $C_k = C_{k+1} = \dots$, and \mathcal{T} is just a finite subdivision of C ; in this case, all the simplices in C_k are stable, and $|K(\mathcal{T})| = |C|$. At the other extreme, if Σ_k is empty for all k , then \mathcal{T} consists of the k^{th} chromatic subdivisions of C for all k ; in this case, no simplices are stable, and $K(\mathcal{T})$ is empty.

5.2. The main result. Let $M \subseteq \mathcal{R}$ be a restricted IIS model on $n+1$ processes. Let \mathcal{T} be a terminating subdivision of the standard n -simplex \mathbf{s} . We say that \mathcal{T} is *admissible* for M if for any run $r \in M$ (viewed as a sequence of simplices $\sigma_0, \sigma_1, \dots$), there exists k such that the geometric realization $|\sigma_k|$ is contained in $|\tau|$ for some stable simplex τ of \mathcal{T} , $\tau \in K(\mathcal{T})$.

Theorem 5.1 (The Generalized Asynchronous Computability Theorem). *A restricted IIS model M solves a task $T = (C, \Delta)$ if and only if there exists a terminating subdivision \mathcal{T} of \mathbf{s} and a chromatic map $\delta : K(\mathcal{T}) \rightarrow C$ such that:*

- (a) \mathcal{T} is admissible for the model M ;
- (b) For any face $\mathbf{t} \subseteq \mathbf{s}$, if σ is a stable simplex of \mathcal{T} such that $|\sigma| \subseteq |\mathbf{t}|$, then $\delta(\sigma) \in \Delta(\mathbf{t})$.

Proof. " \Rightarrow ": Suppose M solves T using a protocol Π . By induction on the recursion that defines $\text{view}(p_i^k, r)$, it is easy to see that the k^{th} view of p_i in r corresponds to a vertex $v \in \text{Chr}^k(\mathbf{s})$ with $\chi(v) = \{p_i\}$.

We now construct \mathcal{T} . We proceed with the standard subdivisions $\text{Chr}^k(\mathbf{s})$ for $k = 0, 1, 2, \dots$, and we examine all runs $r \in M$. At the k^{th} stage we take the inductively constructed C_k , whose vertices are a subset of the vertices of $\text{Chr}^k(\mathbf{s})$. We then terminate those simplices for which Π has given an output: A simplex σ of C_k is included in Σ_k if there exists a run $r \in M$ such that the vertices of σ are of the form $v_i = \text{view}(p_i^k, r)$, and the outputs $\Pi(v_i)$ exist (that is, $v_i \in \text{domain}(\Pi)$). Then Σ_k determines C_{k+1} .

Given a simplex $\sigma \in \Sigma_k$ with vertices v_i , we set $\delta(\sigma)$ be the simplex with vertices $\Pi(v_i)$.

Part (a) (admissibility of \mathcal{T}) follows from the construction: Given any run $r \in M$, pick k such that all the processes infinitely participating in r have produced output at the k^{th} step. If the corresponding simplex $\sigma_k \in \text{Chr}^k \mathbf{s}$ is in C_k , then it is necessarily a stable simplex (because all the processes have output), and we are done. If σ_k is not a simplex of C_k , this means that $|\sigma_k|$ is contained in a simplex of C_k that was terminated before (because some of the processes have produced output at an earlier time).

Part (b) follows from the fact that M solves T using Π .

" \Leftarrow ": Conversely, suppose there exists a terminating subdivision \mathcal{T} and a map $\delta : K(\mathcal{T}) \rightarrow C$ as in the statement of the theorem. We construct a protocol Π by which M solves T . Suppose we have a run $r \in M$, corresponding to a sequence of simplices $\sigma_0 \subseteq \sigma_1 \subseteq \sigma_2 \subseteq \dots$. Since \mathcal{T} is admissible for M , there exists a stable simplex τ such that for $|\sigma_k| \subseteq |\tau|$ for all $k \gg 0$. Given a process $p_i \in \infty\text{-part}(r)$, we can assign it as output value the vertex of $\delta(\tau)$ that has color p_i . Now, p_i may obtain an output value (necessarily the same as before) through another run, at a different step k . We take the minimum over all such k , to obtain the value k_0 needed in the definition of Π . That Π solves T follows from condition (b). \square

6. THE WAIT-FREE MODEL

When $M = WF$, let us see how we can derive the original Asynchronous Computability Theorem of [21]. Indeed, Theorem 5.1 has the following:

Corollary 6.1. *A task $T = (C, \Delta)$ is solvable in the wait-free model if and only if there exists $k \geq 0$ and a chromatic map $\eta : \text{Chr}^k \mathbf{s} \rightarrow C$ such that, for any face $\mathbf{t} \subseteq \mathbf{s}$ and any subsimplex σ of $\text{Chr}^k \mathbf{t} \subset \text{Chr}^k \mathbf{s}$, we have $\eta(\sigma) \in \Delta(\mathbf{t})$.*

Proof. If $\eta : \text{Chr}^k \mathbf{s} \rightarrow C$ exists, solvability of T follows from GACT because $\text{Chr}^k \mathbf{s}$ (with all the vertices terminated at the k^{th} step) is a terminating subdivision that is admissible for WF .

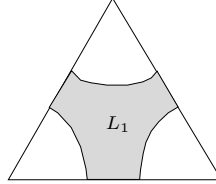
Conversely, suppose that T is wait-free solvable. GACT provides a terminating subdivision \mathcal{T} that is admissible for WF , and a map $\delta : K(\mathcal{T}) \rightarrow C$. For each run $r \in WF = \mathcal{R}$, there is a k such that $|\sigma_k|$ is contained in a stable simplex of \mathcal{T} . Let \mathcal{R}_k be the set of runs for which a specific k works. We have inclusions $\mathcal{R}_0 \subseteq \mathcal{R}_1 \subseteq \mathcal{R}_2 \subseteq \dots$, and each \mathcal{R}_k is open in the topology on \mathcal{R} introduced in Section 4. We know from Lemma 4.1 that the set \mathcal{R} is compact. This implies that there exists k such that $\mathcal{R}_k = \mathcal{R}$. We now define the desired map $\eta : \text{Chr}^k \mathbf{s} \rightarrow C$ by setting $\eta(\sigma) = \delta(\tau)$, where $\tau \in K(\mathcal{T})$ is the minimal simplex with $|\sigma| \subseteq |\tau|$. \square

Remark 6.2. An alternative proof of Corollary 6.1 can be given using König's lemma.

As stated in [21], ACT characterizes solvability in terms of a map from an arbitrary colored subdivision of \mathbf{s} to the output complex. That any colored subdivide simplex can be approximated by $\text{Chr}^k(\mathbf{s})$ for some k large enough is a purely topological result, proved in [21], and which can be used here verbatim.

7. AN EXAMPLE OF GACT IN ACTION

Consider the t -resilient model Res_t from Example 2.2. Let L_t be the affine task with output complex consisting of all the simplices σ in the second chromatic subdivision $\text{Chr}^2 \mathbf{s}$ such that no vertex of σ is on an $(n-t-1)$ -dimensional face of \mathbf{s} . For example, when $n = 2$ and $t = 1$, the output complex for L_1 looks like:



Proposition 7.1. *The task L_t is solvable in the model Res_t .*

Before going into the proof, we need a tool for constructing chromatic maps between two chromatic complexes A and B (subject to some boundary conditions, as required in GACT). In many cases, it is easier to first construct a continuous map $f : |A| \rightarrow |B|$. A standard result in algebraic topology says that after replacing A by a fine enough subdivision, we can deform f into a geometric realization of a simplicial map. Such a map may collapse the dimension of simplices, so it is not always clear how to turn it into a chromatic map. The key property that we need (for the target complex B) is the following:

Definition 7.2 (Definition 4.14 in [21]). A pure n -dimensional subcomplex $L \subseteq \text{Chr}^m \mathbf{s}$ is called *link-connected* if for all simplices $\sigma \in L$, the link of σ in L is $(n - \dim(\sigma) - 2)$ -connected.

For example, the output complex C^{ord} for the total order task on three processes is not link-connected, because the link (in C^{ord}) of a vertex of \mathbf{s} is not connected. On the other hand, the output complex for L_t is link-connected.

The following proposition is an extension of the work of Herlihy and Shavit in [21] to the case of *infinite* chromatic complexes:

Proposition 7.3. *Let M be a restricted IIS model, and \mathcal{T} a terminating subdivision of \mathbf{s} that is admissible for M . Suppose we have a task $T = (C, \Delta)$ such that the complexes $\Delta(\mathbf{t})$ are link-connected, for any $\mathbf{t} \subseteq \mathbf{s}$. If there exists a continuous map $f : |K(\mathcal{T})| \rightarrow |C|$ such that $f(|K(\mathcal{T})| \cap |\mathbf{t}|) \subseteq |\Delta(\mathbf{t})|$ for all \mathbf{t} , then the task T is solvable in M .*

Proof. If \mathcal{T} and \mathcal{T}' are terminating subdivisions of \mathbf{s} , we say that \mathcal{T}' is a *stable refinement* of \mathcal{T} if $|K(\mathcal{T})| = |K(\mathcal{T}')|$, and every simplex of \mathcal{T}' is contained in a simplex of \mathcal{T} . Note that if \mathcal{T} is admissible for a model M , then so is \mathcal{T}' .

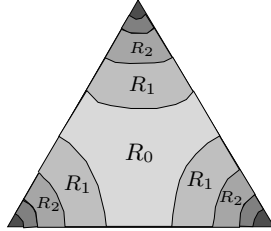
Given the continuous map f , we shall construct a simplicial, chromatic approximation $\delta : K(\mathcal{T}') \rightarrow C$ as needed to apply GACT; here, \mathcal{T}' is a stable refinement of \mathcal{T} .

We first construct a chromatic subdivision K' of $K(\mathcal{T})$, whose vertices are not necessarily in the standard chromatic subdivisions of \mathbf{s} , and a chromatic map $\delta' : K' \rightarrow C$ (an approximation to f) such that δ' is carrier-preserving: $\delta'(\sigma) \in \Delta(\mathbf{t})$ when $|\sigma| \subseteq |\mathbf{t}|$. We do this inductively on $d \geq 0$: For each d , we define the values of δ' on the simplices that are contained in d -dimensional faces of $|\mathbf{s}|$. Suppose we have defined δ' for $d - 1$, and pick a d -dimensional face \mathbf{t} of $|\mathbf{s}|$. The restriction of f to $|K(\mathcal{T})| \cap |\mathbf{t}|$ can be approximated by a simplicial map from a subdivision of $K(\mathcal{T})$, extending the already constructed δ' on the $(d - 1)$ -dimensional boundary. Further, since $\Delta(\mathbf{t})$ is link-connected, it follows from the arguments in [21, Sections 4.5-4.6] that we can arrange for δ' to preserve colors; see in particular Lemmas 4.18, 4.20 and 4.21 in [21].

By the arguments in [21, Theorem 5.29], we can find a sufficiently fine stable refinement \mathcal{T}' of \mathcal{T} and a chromatic, carrier-preserving map $g : K(\mathcal{T}') \rightarrow K'$. We then set $\delta = \delta' \circ g$ and apply Theorem 5.1. \square

Proof of Proposition 7.1. Note that for each $\mathbf{t} \subseteq \mathbf{s}$, the complex $\Delta(\mathbf{t})$ for the task L_t is link-connected. Therefore, it suffices to find a terminating subdivision \mathcal{T} and a continuous map f with the properties required in Proposition 7.3.

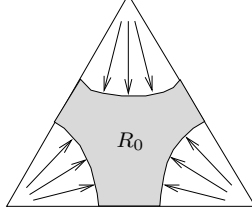
For $n \geq 0$, let $\tilde{R}_n \subset |\mathbf{s}|$ be the union of (the geometric realizations of) all the simplices $\sigma \subset \text{Chr}^{n+2} \mathbf{s}$ such that no vertex of σ is on an $(n - t - 1)$ -dimensional face of \mathbf{s} . Let $R_0 = |L_t|$ and, for $n > 0$, let R_n be the closure of $\tilde{R}_n - \tilde{R}_{n-1}$. The union of all R_n 's is the complement of the $(n - t - 1)$ -skeleton of \mathbf{s} :



The terminating subdivision \mathcal{T} is as follows: It starts with $\Sigma_0 = \Sigma_1 = \emptyset$, so that $C_0 = \mathbf{s}$, $C_1 = \text{Chr}^1 \mathbf{s}$, $C_2 = \text{Chr}^2 \mathbf{s}$. We then let Σ_2 be the subcomplex of $\text{Chr}^2 \mathbf{s}$ supported in the region R_0 . This defines C_2 so that on the complement of R_0 it consists of the simplices in $\text{Chr}^3 \mathbf{s}$. After this, we terminate all the simplices in R_1 . Now on the complement of $R_0 \cup R_1$ we have the fourth chromatic subdivision. We then terminate the simplices in R_2 , and so on. By construction, eventually, every simplex contained in any R_k is stable in this terminating subdivision. Since the affine projection $\pi(\text{Res}_t)$ is contained in the union $|K(\mathcal{T})|$ of all the R_k 's, we deduce that \mathcal{T} is admissible for the model Res_t .

It remains to construct the continuous map $f : |K(\mathcal{T})| \rightarrow |L_t| = R_0$. We let the restriction of f to R_0 be the identity. We then map everything else onto the boundary

$R_0 \cap R_1$ using radial projection away from the $(n - t - 1)$ -skeleton of \mathbf{s} :



Observe that radial projection preserves boundaries, so Proposition 7.3 applies. \square

Remark 7.4. An alternative, operational solution of the challenge task L_t via t -resilient atomic-snapshots was given by the first author in [7].

8. CONCLUDING REMARKS

We presented a version of a generalization of ACT. Other versions may be possible through the relation between simplicial and continuous maps, as well as through defining terminating subdivisions not necessarily with respect to $\text{Chr}^m(\mathbf{s})$, in analogy to the sufficiency condition of ACT [21]. We chose the simplest version which still provides us with the benefit of producing a “topological solution” to the given task.

The main technical challenge we faced was to define and view the IIS model directly, rather than just through the prism of the simulations from the standard (non-iterated) model [3, 13]. This brought forth a coherent view of IIS, as well as exposed the richness of IIS as compared to the standard model.

The generic IIS models considered in this paper are just arbitrary subsets of the various possible interleaving of reads and writes, which is an extension with respect to the previous attempts to model restricted IIS computations [24]. Yet, distributed computing refers also to the availability of one-shot objects, e.g., consensus, k -set agreement, etc. Of course, we can produce a restricted IIS model which is equivalent to having consensus, or any other simple object. The “unifying field question” we contemplate now is whether our framework embedded in a large enough dimension can capture “all of distributed computing,” with respect to terminating computations. In particular, what will be the sets of runs that correspond to the availability of the torus task [14] or of a task from the family of 0-1 exclusion [9]? We know that, for instance, the symmetric task on n processes from [14] can also be formulated as a task on $2n - 1$ processes, hence the increase of dimension holds promise. Our speculation is that any computability question on a conceivable one-shot problem in distributed computing is equivalent to a question on task solvability in a restricted IIS model.

Acknowledgement. We are in debt to Robert F. Brown for helpful discussions in the very early stages of this research.

REFERENCES

- [1] Bowen Alpern and Fred B. Schneider. Defining liveness. *Information Processing Letters*, 21(4):181–185, October 1985.
- [2] Elizabeth Borowsky and Eli Gafni. Immediate atomic snapshots and fast renaming. In *PODC*, pages 41–51, 1993.
- [3] Elizabeth Borowsky and Eli Gafni. A simple algorithmically reasoned characterization of wait-free computation (extended abstract). In *PODC*, pages 189–198, 1997.

- [4] Elizabeth Borowsky, Eli Gafni, Nancy A. Lynch, and Sergio Rajsbaum. The BG distributed simulation algorithm. *Distributed Computing*, 14(3):127–146, 2001.
- [5] Armando Castañeda, Maurice Herlihy, and Sergio Rajsbaum. An equivariance theorem with applications to renaming. In *LATIN*, pages 133–144, 2012.
- [6] Carole Delporte-Gallet, Hugues Fauconnier, Rachid Guerraoui, and Andreas Tielmann. The disagreement power of an adversary. In *DISC*, pages 8–21, 2009.
- [7] Eli Gafni. On the wait-free power of iterated-immediate-snapshots. Unpublished manuscript, online at <http://www.cs.ucla.edu/~eli/eli/wfiis.ps>, 1998.
- [8] Eli Gafni. Round-by-round fault detectors (extended abstract): Unifying synchrony and asynchrony. In *PODC*, 1998.
- [9] Eli Gafni. The 0-1-exclusion families of tasks. In *OPODIS*, pages 246–258, 2008.
- [10] Eli Gafni. Free-for-all execution: Unifying resiliency, set-consensus, and concurrency. Unpublished manuscript, February 2008.
- [11] Eli Gafni. The extended BG-simulation and the characterization of t -resiliency. In *STOC*, pages 85–92, 2009.
- [12] Eli Gafni and Elias Koutsoupias. Three-processor tasks are undecidable. *SIAM J. Comput.*, 28(3):970–983, 1999.
- [13] Eli Gafni and Sergio Rajsbaum. Distributed programming with tasks. In *OPODIS*, pages 205–218, 2010.
- [14] Eli Gafni, Sergio Rajsbaum, and Maurice Herlihy. Subconsensus tasks: Renaming is weaker than set agreement. In *DISC*, pages 329–338, 2006.
- [15] Maurice Herlihy. Wait-free synchronization. *ACM Transactions on Programming Languages and Systems*, 13(1):123–149, January 1991.
- [16] Maurice Herlihy and Sergio Rajsbaum. The decidability of distributed decision tasks (extended abstract). In *STOC*, pages 589–598, 1997.
- [17] Maurice Herlihy and Sergio Rajsbaum. Algebraic spans. *Mathematical Structures in Computer Science*, 10(4):549–573, 2000.
- [18] Maurice Herlihy and Sergio Rajsbaum. Concurrent computing and shellable complexes. In *DISC*, pages 109–123, 2010.
- [19] Maurice Herlihy and Sergio Rajsbaum. The topology of shared-memory adversaries. In *PODC*, 2010.
- [20] Maurice Herlihy and Nir Shavit. The asynchronous computability theorem for t -resilient tasks. In *STOC*, pages 111–120, May 1993.
- [21] Maurice Herlihy and Nir Shavit. The topological structure of asynchronous computability. *Journal of the ACM*, 46(2):858–923, 1999.
- [22] Dmitry N. Kozlov. Chromatic subdivision of a simplicial complex. *Homology, Homotopy and Applications*, 14(1):1–13, 2012.
- [23] Nati Linial. Doing the IIS. Unpublished manuscript, 2010.
- [24] Sergio Rajsbaum, Michel Raynal, and Corentin Travers. The iterated restricted immediate snapshot model. In *COCOON*, pages 487–497, 2008.
- [25] Michael Saks and Fotios Zaharoglou. Wait-free k -set agreement is impossible: The topology of public knowledge. *SIAM J. on Computing*, 29:1449–1483, 2000.